Search Results

Search Results for: **[(replacing AND instruction AND branch) AND (trace AND buffer) <AND>((((identifying AND instruction AND debug) AND (instrumenting AND instruction))))]**
Found **24** of **131,734 searched.**

Search within Results

> Advanced Search

> Search Help/Tips

Sort by: Title Publication Publication Date **Score** Binder

Results 1 - 20 of 24 short listing

Prev Page 1 2 Next Page

1 Partial method compilation using dynamic profile information 80%
John Whaley
ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications October 2001

Volume 36 Issue 11
The traditional tradeoff when performing dynamic compilation is that of fast compilation time versus fast code performance. Most dynamic compilation systems for Java perform selective compilation and/or optimization at a method granularity. This is the not the optimal granularity level. However, compiling at a sub-method granularity is thought to be too complicated to be practical. This paper describes a straightforward technique for performing compilation and optimizations at a finer, sub-metho ...

2 Compiler transformations for high-performance computing 80%
David F. Bacon , Susan L. Graham , Oliver J. Sharp
ACM Computing Surveys (CSUR) December 1994
Volume 26 Issue 4
In the last three decades a large number of compiler transformations for optimizing programs have been implemented. Most optimizations for uniprocessors reduce the number of instructions executed by the program using transformations based on the analysis of scalar quantities and data-flow techniques. In contrast, optimizations for high-performance superscalar, vector, and parallel processors maximize parallelism and memory locality with transformations that rely on tracking the properties o ...

3 Software engineering: applications, practices tools (SE): A portable virtual machine for program debugging and directing 77%
Camil Demetrescu , Irene Finocchi
Proceedings of the 2004 ACM symposium on Applied computing March 2004
Directors are reactive systems that monitor the run-time environment and react to the emitted events. Typical examples of directors are debuggers and tools for program

PORTAL
THE ACM DIGITAL LIBRARY

> home > about > feedback > login
US Patent & Trademark Office

Try the *new* Portal design
Give us your opinion after using it.

Search Results

Search Results for: **[(*) <AND>((madsen)<IN> author)]**
Found **57** of **131,734 searched.**

Search within Results

> Advanced Search
> Search Help/Tips

Sort by: Title Publication Publication Date **Score** Binder

Results 1 - 20 of 57 short listing

Prev Page 1 2 3 Next Page

1 Participatory design in Britain and North America: responses to the "Scandinavian Challenge" 84%
Michael J. Muller , Jeanette L. Blomberg , Kathleen A. Carter , Elizabeth A. Dykstra , Kim Halskov Madsen , Joan Greenbaum
Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology March 1991

2 Strong typing of object-oriented languages revisited 84%
Ole Lehrmann Madsen , Boris Magnusson , Birger Mølier-Pedersen
ACM SIGPLAN Notices , Proceedings of the European conference on object-oriented programming on Object-oriented programming systems, languages, and applications September 1990
Volume 25 Issue 10
This paper is concerned with the relation between subtyping and subclassing and their influence on programming language design. Traditionally subclassing as introduced by Simula has also been used for defining a hierarchical type system. The type system of a language can be characterized as strong or weak and the type checking mechanism as static or dynamic. Parameterized classes in combina ...

3 An algebra for program fragments 84%
Bent Bruun Kristensen , Ole Lehrmann Madsen , Birger Møller-Pedersen , Kristen Nygaard
Proceedings of the ACM SIGPLAN 85 symposium on Language issues in programming environments June 1983
Volume 18 , 20 Issue 6 , 7
Program fragments are described either by strings in the concrete syntax or by constructor applications in the abstract syntax. By defining conversions between these forms, both may be intermixed. Program fragments are constructed by terminal and nonterminal symbols from the grammar and by variables having program fragments as values. Basic operations such as valuetransfer, composition and decomposition are defined for program fragments allowing more complicated operations to be implemented ...

PORTAL THE ACM DIGITAL LIBRARY

> home : > about : > feedback : > login

US Patent & Trademark Office

Try the *new* Portal design

Give us your opinion after using it.

Search Results

Nothing Found

Your search for **[(*) <AND>((lee dersang)<IN> author)]** did not return any results.

You may revise it and try your search again below or click advanced search for more options.

(*) <AND>((lee dersang)<IN> author)

SEARCH [Advanced Search] [Search Help/Tips]

Complete Search Help and Tips

The following characters have specialized meaning:

Special Characters	Description
, ( ) [	These characters end a text token.
= > < !	These characters end a text token because they signify the start of a field operator. (! is special: != ends a token.)
` @ \Q < { [!	These characters signify the start of a delimited token. These are terminated by the end character associated with the start character.

PORTAL
THE ACM DIGITAL LIBRARY

> home : > about : > feedback : > login

US Patent & Trademark Office

Try the *new* Portal design
Give us your opinion after using it.

Search Results

Search Results for: **[(*) <AND>((dawson)<IN> author)]**
Found **56** of **131,734 searched.**

Search within Results

> Advanced Search

> Search Help/Tips

Sort by: Title Publication Publication Date **Score** Binder

Results 1 - 20 of 56 short listing

Prev Page 1 2 3 Next Page

1 Improved effectiveness from a real time LISP garbage collector 82%
Jeffrey L. Dawson
Proceedings of the 1982 ACM symposium on LISP and functional programming
August 1982
This paper describes a real-time garbage collection algorithm for list processing systems. We identify two efficiency problems inherent to real-time garbage collectors, and give some evidence that the proposed algorithm tends to reduce these problems. In a virtual memory implementation, the algorithm restructures the cell storage area more compactly, thus reducing working sets. The algorithm also may provide a more garbage-free storage area at the end of the collection cycle, although this ...

2 Organization of final year projects 82%
Kenneth M. Dawson-Howe
ACM SIGCSE Bulletin September 1996
Volume 28 Issue 3
This paper details a method for the organization of final year computer science projects which has been found to be extremely beneficial both from the point of view of the students and the supervisor. These projects count for 20% of the final degree result in this Department, and are a crucial part of the development of the student.The model proposed for the organization of the projects is one in which the students initially work in a group, co-operatively developing a basic platform on which th ...

3 Automatic submission and administration of programming assignments 82%
Kenneth M. Dawson-Howe
ACM SIGCSE Bulletin June 1996
Volume 28 Issue 2
This paper details a system to assist with the evaluation and administration of student assignments. In order to help with the evaluation of program execution, the system automatically compiles and executes the program while logging a copy of the session. This log, together with the code and documentation is then bundled into an e-mail which is sent to the course controller.The course controller automatically processes the e-mail, verifying it's authenticity, and sends an acknowledgement back to ...